HOLA: HADAMARD HIGHER-ORDER ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The attention mechanism is an important reason for the success of transformers. It relies on computing pairwise relations between a query and all key tokens and applying softmax normalization. The expressiveness of softmax stems from the exponential kernel, which, however, causes quadratic computational complexity. Linear attention replaces the exponential by a single inner product of feature maps. In terms of the Taylor-series expansion of the exponential function, this amounts to a first-order approximation. Prior work has shown that the second-order term can also be computed in linear time.

In this paper, we introduce Hadamard Higher-Order Linear Attention (HOLA), which extends this construction in three directions. It supports efficient factorizations of arbitrary-order terms rather than only the second-order term. The shared symmetric feature map used by prior works is replaced with distinct maps for each factor in the higher-order Hadamard product. This relaxes a low-rank constraint on the state matrix. HOLA amounts to standard linear attention at order one, includes the prior second-order construction as an exact special case, and is compatible with existing mechanisms such as gating, decay, and delta-rule updates.

1 INTRODUCTION

The transformer architecture has demonstrated remarkable success across a wide range of domains, for instance, question answering Sima et al. (2024), reasoning Wei et al. (2022), and even tasks such as 3D reconstruction Wang et al. (2024). The attention mechanism is central to the success of transformers, but its cost grows quadratically with sequence length. While recent advances Dao et al. (2022); Dao (2023); Shah et al. (2024) have removed the quadratic *memory* cost, the *computational* cost remains quadratic. This is the binding constraint in long-context regimes such as book-length question answering and video generation, where sequences easily reach 10^4–10^5 tokens.

Linear attention Katharopoulos et al. (2020); Schlag et al. (2021) replaces the exponential function used by the standard attention mechanism by a separable kernel, which yields linear cost and a constant-size recurrent state. The price is expressiveness: the resulting state matrix has a low rank constraint, whereas softmax attention maps are empirically high-rank with extreme peaks.

Prior methods can be partitioned into several categories based on the aspect they address. The first is a better or learned feature map Zhang et al. (2024b;a); Qin et al. (2022); Meng et al. (2025). The second is the *state update rule*: decay, gating and the delta rule Schlag et al. (2021); Yang et al. (2024; 2025a); Dao and Gu (2024), or sparsity and hybridization with exact attention Zhang et al. (2025); Yuan et al. (2025); Lu et al. (2025). Both leave the attention score *bilinear* in the transformed pair, so have *polynomial degree* 1. Softmax attention, on the other hand, differs precisely with respect to this, since the degrees of the monomials of the Taylor series expansion of the exponential function increase without bound.

The order-two term of this expansion admits an efficient factorization. This observation is due to BASED Arora et al. (2024) and, independently, TaylorShift Nauen et al. (2024); PolySketchFormer Kacham et al. (2024) extends a related factorization to higher degrees using sketching. Both prior constructions share two properties that we remove. First, they compute a Hadamard product of a single score matrix with itself. The score matrix is built from tied query and key maps: the same map is applied to both terms of the Hadamard product for BASED Arora et al. (2024) and TaylorShift Nauen et al. (2024). While PolySketchFormer Kacham et al. (2024) uses different feature maps for queries and keys, the maps are still tied among the factors of the Hadamard product. Second,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

they lift to the full per-head width, so the order-two lift has complexity quadratic in the per-head width and does not extend past order two in practice as tertic and higher products are intractable for typical head dimensions. The authors of PolySketchFormer Kacham et al. (2024) propose an approximation to higher-order products.

Although computing the Hadamard product of a score matrix with itself should significantly relax the low-rank constraint on the state matrix, we will point out that using *shared* feature maps induces a symmetry which in turn causes a much tighter low-rank constraint. We therefore propose to use independent feature maps at each factor of the Hadamard product. For efficiency, we propose to use feature maps of dimensions smaller than the head dimension. By using efficient CUDA kernels, we demonstrate that the exponential can be efficiently approximated with *all* terms of the Taylor expansion up to degree-3. Even higher-order approximations might become feasible at longer sequence lengths. These CUDA kernels were created using commercial LLMs. Both higher degrees and asymmetric feature maps make the linear attention mechanism more expressive. We evaluate the proposed algorithm on challenging long-range associative retrieval and video generation with sequence lengths of several 10k tokens.

2 Related Works

Linear attention has been proposed to solve the quadratic memory requirement and computation complexity that standard softmax-based attention has Shen et al. (2024); Katharopoulos et al. (2020). It has been noted that algorithms relying on linear attention often perform poorly compared to algorithms using standard attention. The authors of Schlag et al. (2021) point out that linear attention is equivalent to fast weight programmers Schmidhuber (1992). The same authors also propose to use a decay factor to hinder retention of irrelevant information over a sequence via the Delta rule. Since the latency of this algorithm increases with the sequence length, a chunked version of this algorithm has been proposed in Qiu et al. (2025). For LLM training, a similar idea has been proposed in Sun et al. (2023).

Peng et al. (2021); Choromanski et al. (2022); Qin et al. (2022); Meng et al. (2025) propose how to better approximate the exponential function in the context of linear attention. The low-rank constraint inherent to linear attention is a likely cause of the poor performance of linear attention. A solution has been proposed in Fan et al. (2025). Other approaches have been proposed as well, for instance based on maximal coding rate reduction Wu et al. (2024) or locality sensitive hashing Han et al. (2023).

A different approach to reducing the quadratic complexity of standard attention is State Space Methods (SSMs). They model the temporal change by a set of first-order differential equations which are recurrently applied to the input sequence. They can perform well on long-range tasks Gu et al. (2022) and can match or even transformers in some tasks Gu and Dao (2024). Connections between linear attention and SSMs were revealed in Dao and Gu (2024). Using the delta rule as in Schlag et al. (2021) can also benefit SSMs Yang et al. (2025a). Drawing on ideas from SSMs, the authors of Egorov et al. (2025) improve a standard attention attention mechanism without sacrificing performance.

Linear attention has been used to improve diffusion networks, for instance in Xie et al. (2024); Zhu et al. (2024) for images. Linear attention improved by the delta rule has also been used in diffusion algorithms for video generation Chen et al. (2025); Team et al. (2025). The authors of Ye et al. (2025) improve standard attention by performing it twice and subtracting the two results.

Recently, Zhang et al. (2025) and Ghafoorian et al. (2025) combined softmax attention with linear attention for video diffusion. This idea is to model large values in the attention matrix by softmax, yet smaller ones by linear attention. While these hybrid methods offer good performance, they require dynamical indexing and are not softmax-free. Dynamical indexing means to directly selecting one or multiple individual values in a tensor. This operation is not supported by some low-power compute units.

3 Background

Softmax attention. Let $\boldsymbol{X} = \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top\right] \in \mathbb{R}^{N \times d}$ be N tokens, mapped to queries, keys and values Q, K, V by learned projections W_q, W_k, W_v. Attention forms a convex combination of

Table 1: Notation. Throughout, $\odot$ is the Hadamard (element-wise) product, $\otimes$ the Kronecker product, and $*$ the row-wise Kronecker (face-splitting) product, $(A * B)_{i,:} = A_{i,:} \otimes B_{i,:}$. For $v \in \mathbb{R}^n$, $v^{\otimes F} = v \otimes \cdots \otimes v$ (F copies) denotes the F-fold Kronecker power, with entries $(v^{\otimes F})_{a_1 \ldots a_F} = \prod_f v_{a_f}$.

N, d	sequence length, channels per head
$Q, K, V \in \mathbb{R}^{N \times d}$	queries, keys, values
F	interaction order (number of factors)
d_ϕ	width of each feature map, $d_\phi \leq d$
$\phi_{q_f}, \phi_{k_f} : \mathbb{R}^d \to \mathbb{R}^{d_\phi}_{\geq 0}$	feature maps of factor $f = 1, \ldots, F$
$\tilde{q}_i, \tilde{k}_j$	lifted features, dimension d_ϕ^F
S, z	state and normaliser
$\mathrm{Sym}^F(\mathbb{R}^{d_\phi})$	symmetric subspace, $\dim = \binom{d_\phi + F - 1}{F}$

values (Vaswani et al., 2017),

$$\mathrm{softmax}\Big(\tfrac{1}{\sqrt{d}}\, QK^\top\Big)\, V, \tag{1}$$

at $\mathcal{O}(N^2 d)$ cost. Implementations optimized for efficiency (Dao et al., 2022; Dao, 2023; Shah et al., 2024) remove the quadratic *memory* cost but not the quadratic *compute*.

Linear attention. Linear attention (Katharopoulos et al., 2020; Schlag et al., 2021) replaces the exponential by a separable kernel,

$$\exp\langle q_i, k_j \rangle \;\approx\; \langle \phi(q_i), \phi(k_j) \rangle, \tag{2}$$

with a *feature map* $\phi : \mathbb{R}^d \to \mathbb{R}^{d_\phi}$. Separability permits reassociating the products,

$$\phi(Q)\,\big(\phi(K)^\top V\big) \;=\; \phi(Q)\, C, \tag{3}$$

so the $N \times N$ score matrix is never formed: cost drops to $\mathcal{O}(N d_\phi d)$ and the $d_\phi \times d$ *state* (or context) C is all that must be stored, which also yields an $\mathcal{O}(1)$-per-step recurrent form. The price is expressiveness. Since $\mathrm{rank}(\phi(Q)C) \leq \min(\mathrm{rank}\,\phi(Q), \mathrm{rank}\, C) \leq d_\phi$, the implied score matrix is confined to rank d_ϕ, whereas measured softmax maps are high-rank (Zhang et al., 2025) and strongly peaked (Fan et al., 2025); equivalently, the output is a low-degree rational function of the inputs (Letourneau et al., 2024).

Existing algorithms address two problems. The first is the *feature map*: random or orthogonal Fourier features (Peng et al., 2021; Choromanski et al., 2020), trigonometric surrogates (Qin et al., 2022; Meng et al., 2025), and maps trained to mimic softmax attention (Zhang et al., 2024b;a); the last also *untie* the query and key maps, using separate ϕ_q and ϕ_k. The second is the *state update rule*: decay, gating and the delta rule (Schlag et al., 2021; Yang et al., 2024; 2025a; Dao and Gu, 2024), or sparsity and hybridisation with exact attention (Zhang et al., 2025; Yuan et al., 2025; Lu et al., 2025).

Interaction order: a third, orthogonal approach. The algorithms in the paragraph above leave the score *bilinear* in the transformed pair: however elaborate ϕ is, $\phi(q_i)^\top \phi(k_j)$ is a single inner product. Softmax attention differs precisely here, since

$$\exp\big(q_i^\top k_j\big) \;=\; \sum_{m \geq 0}^{\infty} \frac{\big(q_i^\top k_j\big)^m}{m!} \tag{4}$$

contains terms of every degree in the *pair*. We call F the *interaction order* of a score that is a degree-F monomial in the pair, i.e. a product of F separable terms; linear attention has order one for *any* feature map, and equation 4 has unbounded order.

The degree-two term of equation 4 admits an exact linear-cost factorization, because $(q^\top k)^2 = (q \otimes q)^\top (k \otimes k)$ is an inner product in a lifted space. This is the construction of Based (Arora et al., 2024) and, independently, TaylorShift (Nauen et al., 2024). Both order-two constructions apply a single, shared map to queries and keys, so all factors of the product are identical, and both lift the full per-head width, so the state has memory complexity $\mathcal{O}(d_\phi d)$ and higher degrees are out of reach. Section 4 starts from exactly this factorization.

4 HIGHER-ORDER LINEAR ATTENTION

4.1 FROM THE EXPONENTIAL SERIES TO ORDER-F SCORES

Linear attention truncates equation 4 at interaction order one. We ask instead for the cheapest exact realisation of the higher-order terms. Let $\phi_{q_f}, \phi_{k_f} : \mathbb{R}^d \to \mathbb{R}^{d_\phi}_{\geq 0}$, $f = 1, \ldots, F$, be learned feature maps and define the *order-F* score matrix as a Hadamard product of F separable score matrices,

$$A^{(F)} \;=\; \bigodot_{f=1}^{F} \Big(\phi_{q_f}(Q)\, \phi_{k_f}(K)^\top \Big) \;\in\; \mathbb{R}^{N \times N}, \qquad A^{(F)}_{ij} = \prod_{f=1}^{F} \big\langle \phi_{q_f}(q_i),\, \phi_{k_f}(k_j) \big\rangle. \tag{5}$$

Nonnegative maps keep every entry nonnegative, so equation 5 can be row-normalised exactly as in linear attention. Setting $F = 1$ recovers linear attention; setting $\phi_{q_f} = \phi_{k_f} = \phi$ for all f yields $\big(\phi(Q)\phi(K)^\top\big)^{\odot F}$ used at $F = 2$ by Based (Arora et al., 2024) and TaylorShift (Nauen et al., 2024). Our construction differs from those in three respects: the order F is arbitrary; the factors are *untied*, i.e. each carries its own maps rather than repeating one score matrix; and the width d_ϕ is chosen *below* the per-head channel count d, which is what makes $F \geq 3$ affordable at all.

4.2 LINEAR-TIME EVALUATION

Naïvely evaluating equation 5 directly has memory and computational complexity $\mathcal{O}(N^2)$. The Hadamard product of Gram matrices is itself a Gram matrix in a lifted space, by the face-splitting (row-wise Kronecker) identity $(AB^\top) \odot (CD^\top) = (A * C)(B * D)^\top$. Applying it $F - 1$ times to equation 5 gives $A^{(F)}_{ij} = \tilde{q}_i^\top \tilde{k}_j$ with the lifted features

$$\tilde{q}_i = \phi_{q_1}(q_i) \otimes \cdots \otimes \phi_{q_F}(q_i) \in \mathbb{R}^{d_\phi^F}, \qquad \tilde{k}_j = \phi_{k_1}(k_j) \otimes \cdots \otimes \phi_{k_F}(k_j) \in \mathbb{R}^{d_\phi^F}. \tag{6}$$

Notice that $\tilde{q}_i$ and $\tilde{k}_j$ are degree-F polynomials of q_i and k_j. By equation 6, equation 5 may therefore be expressed by a single state, in $\mathcal{O}(N d_\phi^F d)$ time and $\mathcal{O}(d_\phi^F d)$ memory,

$$S = \sum_j \tilde{k}_j v_j^\top \in \mathbb{R}^{d_\phi^F \times d}, \qquad z = \sum_j \tilde{k}_j \in \mathbb{R}^{d_\phi^F}, \qquad o_i = \frac{\tilde{q}_i^\top S}{\tilde{q}_i^\top z}. \tag{7}$$

Restricting both sums to $j \leq i$ gives causal attention; multiplying the summands by decay factors gives gated variants; and equation 7 is a recurrence, so it also yields $\mathcal{O}(1)$-per-step decoding with the state as the carried quantity. Because equation 7 is ordinary linear attention on the lifted key $\tilde{k}$, gating, decay and the delta rule (Schlag et al., 2021; Yang et al., 2024; 2025a) readily apply with the key dimension d replaced by d_ϕ^F; Sec. 5 verifies this by upgrading a gated delta network.

4.3 SHARPENING AND RANK CONSTRAINT

Order acts on the attention distribution in an exactly characterisable way. Fix a row i, assume tied factors $\phi_{k_1} = \cdots = \phi_{k_F} = \phi_k$ and write $b_j = \langle \phi_q(q_i), \phi_k(k_j) \rangle > 0$. Then $A^{(F)}_{ij} = b_j^F$, so with $z_j = \log b_j$ the normalised weights are

$$p_j^{(F)} = \frac{b_j^F}{\sum_\ell b_\ell^F} = \mathrm{softmax}(Fz)_j = \mathrm{softmax}_T(z)_j, \qquad T = \tfrac{1}{F}, \tag{8}$$

and every finite pairwise log-odds is scaled exactly by F, $\log\big(p_j^{(F)} / p_{j'}^{(F)}\big) = F\,(z_j - z_{j'})$. Raising the interaction order is therefore *exactly* equivalent to lowering a softmax temperature on the log-scores: the ranking of keys is preserved, while contrasts are amplified linearly in F, and the distribution concentrates on the arg-max as F grows. This is the sense in which order-F attention is peaked like softmax and unlike order-one linear attention, and it requires no approximation argument. With untied factors the corresponding statement is an inequality: for factor scores $s_f(j) \in [0, 1]$ one has $A^{(F)}_{ij} = \prod_f s_f(j) \leq \min_f s_f(j)$, so a key scores highly only if it scores highly in *every* factor simultaneously – a product-of-experts effect.

The Hadamard product relaxes the low-rank constraint of equation 3, since $\operatorname{rank}(A \odot B) \leq \operatorname{rank}(A)\operatorname{rank}(B)$ and hence

$$\operatorname{rank}(A^{(F)}) \leq d_\phi^F. \tag{9}$$

That bound is correct but loose, and the following makes the attainable ceiling precise.

Proposition 4.1 (Rank ceiling under tied query maps)**.** *Let the query map be shared across factors, $\phi_{q_1} = \cdots = \phi_{q_F} = \phi_q$, and let the key maps $\phi_{k_1}, \ldots, \phi_{k_F}$ be arbitrary. Then*

$$\operatorname{rank}\big(A^{(F)}\big) \;\leq\; \min\left(N,\; \binom{d_\phi + F - 1}{F}\right), \tag{10}$$

irrespective of whether the key maps are tied or untied. Attaining $\min(N, d_\phi^F)$ requires untying the query *maps.*

Proof sketch. With a shared query map, $\tilde{q}_i = \phi_q(q_i)^{\otimes F}$ has entries $\prod_f \phi_q(q_i)_{a_f}$, which are invariant under permutations of $(a_1, \ldots, a_F)$; hence $\tilde{q}_i \in \operatorname{Sym}^F(\mathbb{R}^{d_\phi})$, a subspace of dimension $\binom{d_\phi+F-1}{F}$ (the number of degree-F monomials in d_ϕ variables). Since $A^{(F)} = \tilde{Q}\tilde{K}^\top$, we have $\operatorname{rank}(A^{(F)}) \leq \operatorname{rank}(\tilde{Q}) \leq \dim \operatorname{Sym}^F$. Moreover, writing P for the orthogonal projector onto Sym^F, $\langle \tilde{q}, \tilde{k} \rangle = \langle \tilde{q}, P\tilde{k} \rangle$, so the score sees only the symmetric part of the key lift and any asymmetry induced by untied key maps is annihilated. □

This implies that a feature map that is tied among the different Hadamard factors as done by prior works (Arora et al., 2024; Nauen et al., 2024; Kacham et al., 2024) causes a more restrictive constraint on the rank. Conversely, using untied query and key maps admits the more relaxed *upper bound* of the rank in equation 9.

5 Experiments

5.1 Experimental Overview

We evaluate the proposed higher-order attention mechanisms in two distinct domains. First, we study long-context associate retrieval by replacing the attention layers of Qwen2-1.5B and evaluating the resulting models on natural-language modeling, long-range associative retrieval. Second, we integrate the proposed mechanism into the Wan2.1-1.3B video diffusion model and evaluate its ability to replace softmax attention in large-scale video generation. These experiments test whether the proposed mechanism transfers across modalities and whether its computational advantages can be obtained without substantially compromising generation quality. Detailed dataset construction procedures, architecture diagrams, and additional training specifications are provided in the appendix.

5.2 Long-Context Associative Retrieval

5.2.1 Models and Training

Model variants: We use Qwen2.5-1.5B-Instruct as the base language model. Since the softmax-based attention has accesss to all tokens, we evaluate this baseline in zero-shot mode. Furthermore, we replace its attention layers with the several linear attention models as well as the asymmetric Hadamard Linear Attention mechanisms considered in this work. Specifically, we evaluate linear attention, sink sliding-window attention (SWA) Jolicoeur-Martineau et al. (2026), chunked Gated Delta Net (GDN) Schlag et al. (2021); Yang et al. (2024; 2025a), symmetric Hadamard Linear Attention with two factors (HLA-F2), asymmetric HLA-F2, a model with all Taylor-series terms up to third order, HOLA-F3, and a hybrid architecture combining GDN with asymmetric HLA-F2, denoted GDN-asym-HLA-F2. Notice that symmetric HLA-F2 is similar to the model proposed by Arora et al. (2024); Nauen et al. (2024) but unties the key feature maps. HOLA-F3 differs from the model proposed by Arora et al. (2024) because it uses different feature maps for each term of the Taylor-series, though it still is *symmetric* since the query feature maps are tied for the second and third order terms. Unless stated otherwise, we fine-tune all model parameters. For SWA, full-parameter fine-tuning did not yield a stable training; we therefore train rank-16 LoRA adapters while keeping the remaining parameters fixed. Detailed definitions and architecture diagrams for all model variants are provided in Appendix F.

Post-surgery recovery: Replacing the pretrained attention layers causes an immediate degradation in language-modeling performance because the newly introduced sequence-mixing modules are not aligned with the pretrained representations. We therefore perform a recovery phase before long-context fine-tuning. Each modified model is trained on next-token prediction using Wiki101 for $16{,}000$ optimization steps with a batch size of 8. During this phase, we use a learning rate of 10^{-3} for the newly introduced sequence-mixing modules and 10^{-4} for all remaining trainable parameters.

Long-context curriculum. Following recovery training, we fine-tune the models using a three-stage curriculum with sequence lengths of 8k, 16k, and 32k tokens. The learning rates for the three stages are 10^{-4}, 5×10^{-5}, and 2.5×10^{-5}, respectively, for all models but the GDN variants which required lower learning rates for stable training (5×10^{-5}, 2.5×10^{-5} and 1.0×10^{-5}). At every stage, the sequence-mixing modules use a learning rate ten times larger than that of the remaining trainable parameters.

The curriculum combines BookSum-based text summarization, open-vocabulary retrieval and copy tasks, and raw-prose continuation. BookSum examples train the models on long-form natural-language understanding, whereas the constructed tasks provide controlled signals for copying and retrieving information over long distances. See sec. **??** for more information about dataset and task creation. All tasks use a causal language-modeling objective, with loss computed only over assistant-response tokens. User instructions and source contexts are masked.

The loss weights for BookSum instruction following, associative retrieval and copy, and prose continuation are 0.25, 0.75, and 1.5, respectively. Since the effective contribution of a task depends on both its loss weight and its sampling frequency, we report the complete data mixture and the corresponding sampling proportions in appendix **??**. The appendix also describes the BookSum preprocessing pipeline, the generation of synthetic retrieval examples, and the deterministic resampling procedure used to prevent memorization across training epochs.

5.2.2 Evaluation Protocol

We evaluate the language models along two complementary dimensions. First, we measure long-range retrieval using multi-query associative recall (MQAR) tasks. Second, we evaluate natural-language modeling on held-out BookSum examples.

MQAR evaluates the retrieval of multiple arbitrary facts inserted into the input. MQAR increases the difficulty over needle-in-a-haystack (NIAH) by varying the number of competing key-value associations. Preliminary experiments indicated that a straightforward construction of the retrieval data allowed models to exploit unintended shortcuts. We therefore substantially expanded the underlying sampling space to minimize the feasibility of shortcut-based solutions. We additionally evaluate generic copying as a lower-complexity control task. For all controlled tasks, the model generates its answer autoregressively, and we report exact-match accuracy.

We use the final checkpoints and evaluate them on the held-out validation data at sequence lengths of 8k, 16k and 32k tokens.

5.2.3 Main Results

Table 2 compares the sequence-mixing mechanisms after the complete long-context curriculum. We organize the evaluation around four questions:

1. Can the modified models recover the language-modeling capabilities disrupted by attention replacement?
2. How does each mechanism scale as context length and retrieval distance increase?
3. Does asymmetric HLA improve robustness to multiple competing associations?
4. Does asymmetric HLA improve performance of other linear attention variants such as Gated Delta Nets?

The zero-shot Qwen2.5-1.5B baseline achieves no measurable retrieval accuracy at any evaluated sequence length, highlighting the difficulty of the MQAR task under this evaluation setting. Similarly, SWA-lora16 and symmetric HLA-F2 fail to retrieve the correct association. Increasing the interaction

Table 2: Results on MQAR across different sequence lengths. NTP denotes next-token prediciotn loss, MQAR the retrieval loss (cross-entropy), ACC the accuracy in percent.

	8k			16k			32k		
Model	NTP↓	MQAR↓	ACC↑	NTP↓	MQAR↓	ACC↑	NTP↓	MQAR↓	ACC↑
Qwen2.5-1.5B	2.41	1.05	0.0	1.85	1.02	0.0	1.30	1.00	0.0
SWA-lora16	3.59	2.76	0.0	3.64	2.89	0.0	3.60	2.92	0.0
GDN	6.43	0.32	39.1	6.32	0.37	43.4	6.28	0.42	40.6
HLA-F2-sym	6.27	2.34	0.0	6.13	2.34	0.0	5.98	2.32	0.0
HOLA-F3-sym	5.03	1.03	8.7	5.12	0.84	19.4	5.31	0.92	24.7
HLA-F2-asym	6.17	1.03	26.3	6.03	0.99	27.8	5.77	1.13	27.5
GDN-HLA-F2-asym	6.83	0.07	**84.1**	6.74	0.05	**82.8**	6.67	0.08	**79.4**

order improves retrieval: HOLA-F3-sym reaches accuracies of 8.7%, 19.4%, and 24.7% at sequence lengths of 8k, 16k, and 32k, respectively. However, relaxing the symmetry constraint is more effective. Despite using only second-order interactions, HLA-F2-asym consistently outperforms HLA-F2-sym and attains accuracies between 26.3% and 27.8%. Its performance also remains approximately stable across sequence lengths, suggesting that its retrieval capability transfers across the evaluated context-length range.

These retrieval gains are accompanied by weaker next-token prediction performance. All attention-replacement models have substantially higher NTP loss than the Qwen2.5-1.5B baseline. Thus, although higher-order and asymmetric interactions improve associative retrieval, they do not by themselves recover the language-modeling quality of the pretrained Transformer. HOLA-F3-sym additionally exhibits a non-monotonic dependence on sequence length: its retrieval accuracy increases with sequence length, while its NTP loss is lowest at 8k and highest at 32k. This behavior suggests that retrieval accuracy is not determined by sequence length alone and may also depend on the distribution of retrieval instances encountered during the curriculum.

GDN provides a stronger retrieval baseline than the standalone HLA variants, achieving accuracies of 39.1%-43.4%. The best overall results are obtained by GDN-HLA-F2-asym, which replaces GDN's linear-attention component with asymmetric HLA-F2. The hybrid reaches accuracies of 84.1%, 82.8%, and 79.4% at 8k, 16k, and 32k, respectively, corresponding to absolute improvements of 45.0, 39.4, and 38.8 percentage points over standard GDN. It also achieves the lowest MQAR loss at every sequence length. These results indicate that the asymmetric HLA mechanism and the gating and state-update structure of GDN provide complementary benefits. At the same time, the hybrid has the highest NTP loss among the evaluated models, revealing a pronounced trade-off between associative retrieval and next-token prediction performance in the current training setup.

5.2.4 Ablation Studies

We conduct ablations to isolate the contributions of interaction order, symmetry and architectural hybridization.

We compare symmetric HLA-F2, asymmetric HLA-F2, and HOLA-F3 under an otherwise identical language-model training setup, thereby isolating the effects of the symmetry constraint and interaction order from differences in data and optimization. HOLA-F3 provides a higher-order approximation of the Taylor expansion and has more parameters than HLA-F2. Nevertheless, asymmetric HLA-F2 consistently outperforms HOLA-F3. This comparison suggests that relaxing the symmetry constraint is more consequential than increasing the interaction order from two to three. In particular, the results support our hypothesis that symmetry restricts the effective rank of the state matrix, limiting model capacity despite the more accurate higher-order approximation provided by HOLA-F3.

We compare GDN, asymmetric HLA-F2, and GDN-asym-HLA-F2 to assess whether the two mechanisms provide complementary benefits. Specifically, we examine whether replacing the linear-attention component of GDN with asymmetric HLA-F2 improves robustness to competing associations without compromising natural-language modeling performance. While GDN and asymmetric HLA-F2 achieve comparable performance in isolation, their hybrid, GDN-asym-HLA-F2, performs

best overall. These results indicate that asymmetric HLA-F2 can serve as an effective replacement for linear attention within GDN and that the two mechanisms provide complementary modeling capabilities.

5.3 Video Generation

5.3.1 Models and Training

Training data. We fine-tune the video-generation models on a subset of approximately $350{,}000$ videos from the data used by OpenSoraPlan Lin et al. (2024). We augment this dataset with approximately $100{,}000$ synthetic videos generated using Wan2.1-14B.

Base model and architecture variants. We integrate the proposed attention mechanism into the Wan2.1-1.3B video diffusion model Wan et al. (2025). We retain the native output configuration of the base model, corresponding to 81 frames at a spatial resolution of 480×832 pixels.

For the HLA-F2 variant, we replace softmax attention with non-causal HLA-F2 in several of the model's 30 attention layers. The remaining layers retain softmax attention. These layers are selected based on an analysis of pre-softmax attention-score magnitudes: we preserve softmax attention in the nine layers that exhibit the most pronounced extreme scores across a set of representative inputs. More information about the different models can be found in the appendix.

Training procedure. Training consists of two phases. In the first phase, we freeze the pretrained backbone and train only the newly introduced HLA-F2 or linear attention modules for $10{,}000$ optimization steps. In the second phase, we unfreeze the complete model and continue training all parameters. See the appendix for information how many training steps was used for each model.

We use a learning rate of 1×10^{-4} in all phases and train on eight NVIDIA H100 GPUs with a global batch size of 32. We use $1{,}000$ warmup steps in the full-model fine-tuning phase.

Results are shown in fig. 1. See the appendix for the complete optimization setting and more qualitative and also quantitative results.



Figure 1: Results generated using the proposed video diffusion transformer with 10 layers HLA-F2, see the appendix for more details.

5.4 Computational Efficiency

We evaluate the computational efficiency of asymmetric HLA-F2 against FlashAttention-2 causal attention at batch size 1, number of heads 12, sequence lengths of 16k, 32k, and 64k, using $d_k \in 16, 64$

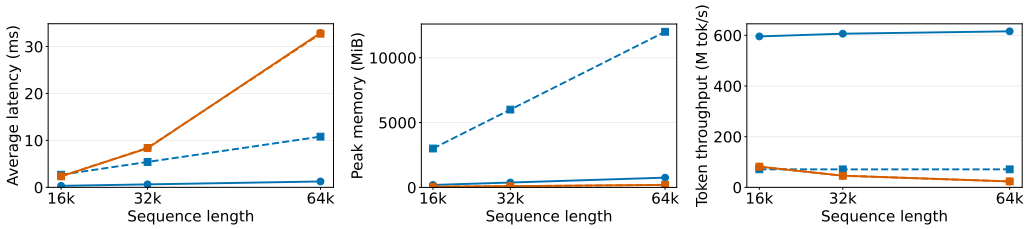

Figure 2: Computational efficiency of asymmetric HLA-F2 and FlashAttention. Forward latency (left), peak memory (middle), and token throughput (right) for sequence lengths from 16k to 64k, with $d_k = 16$ (solid) and $d_k = 64$ (dashed). HLA-F2 exhibits approximately linear latency growth and near-constant throughput, whereas FlashAttention shows quadratic latency growth and decreasing throughput. FlashAttention, however, retains a substantially smaller memory footprint.

and $d_v = 128$. We report forward-pass latency, peak accelerator memory, and token throughput under otherwise fixed benchmark settings.

Figure 2 shows distinct scaling behavior between the two mechanisms. The latency of asymmetric HLA-F2 grows approximately linearly with sequence length, while FlashAttention exhibits the quadratic growth expected from softmax attention. Consequently, the latency advantage of HLA-F2 becomes increasingly pronounced at longer sequences. For $d_k = 16$, HLA-F2 requires 0.32ms, 0.63ms, and 1.25ms at 16k, 32k, and 64k tokens, respectively, compared with 2.35, 8.33, and 32.90 ms for FlashAttention. At 64k tokens, this corresponds to a $26.3\times$ latency reduction.

The two approaches exhibit the opposite trade-off in memory. FlashAttention maintains a small memory footprint that grows only modestly with sequence length, whereas the HLA-F2 CUDA kernel exhibits linear memory growth. This dependence is particularly pronounced for $d_k = 64$. In throughput, HLA-F2 remains approximately constant as sequence length increases, achieving roughly 596-616M tokens/s for $d_k = 16$ and about 71M tokens/s for $d_k = 64$. FlashAttention throughput, in contrast, decreases with increasing sequence length, from approximately 82 M tokens/s at 16k to 23M tokens/s at 64k. These results demonstrate the expected compute-scaling advantage of HLA-F2 at long sequence lengths, while also exposing its higher memory cost relative to FlashAttention.

6 Conclusions

We introduced a generalization of Hadamard Linear Attention (HLA) that relaxes an implicit symmetry constraint, thereby reducing its restriction on the rank of the state matrix while retaining linear complexity in the sequence length. Asymmetric HLA improved associative retrieval more effectively than increasing the interaction order, and its integration into a chunked Gated Delta Network achieved the best results on multi-query associative retrieval (MQAR). Our video-diffusion experiments further demonstrate the applicability of the non-causal formulation. However, the language-modeling results reveal a trade-off between retrieval performance and next-token prediction quality, motivating further work on architectures that preserve both capabilities.

p

References

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.18668.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, Haozhe Liu, Hongwei Yi, Hao Zhang, Muyang Li, Yukang Chen, Han Cai, Sanja Fidler, Ping Luo, Song Han, and Enze Xie. Sana-video: Efficient video generation with block linear diffusion transformer, 2025.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. *arXiv*, 2020.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv*, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality, 2024.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Evgenii Egorov, Hanno Ackermann, Markus Nagel, and Hong Cai. Myosotis: structured computation for attention like layer, 2025.

Qihang Fan, Huaibo Huang, and Ran He. Breaking the low-rank dilemma of linear attention, 2025.

Mohsen Ghafoorian, Denis Korzhenkov, and Amirhossein Habibian. Attention surgery: An efficient recipe to linearize your video diffusion transformer, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion, 2024.

Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P. Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time, 2023.

Jiancheng Huang, Gengwei Zhang, Zequn Jie, Siyu Jiao, Yinlong Qian, Ling Chen, Yunchao Wei, and Lin Ma. M4v: Multi-modal mamba for text-to-video generation, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models. In *Computer Vision and Pattern Recognition (CVPR)*, 2024.

Takashi Isobe, He Cui, Dong Zhou, Mengmeng Ge, Dong Li, and Emad Barsoum. Amd-hummingbird: Towards an efficient text-to-video model, 2025.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling, 2025.

Alexia Jolicoeur-Martineau, Rhea Sanjay Sukthanker, Pashmina Cameron, and Emy Gervais. Sliding-window beats linear attention. 2026. URL `https://arxiv.org/abs/2608.28444`.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. Polysketchformer: Fast transformers via sketching polynomial kernels. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2310.01655.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 5156–5165, 2020.

Pierre-David Letourneau, Manish Kumar Singh, Hsin-Pai Cheng, Shizhong Han, Yunxiao Shi, Dalton Jones, Matthew Harper Langston, Hong Cai, and Fatih Porikli. Padre: A unifying polynomial attention drop-in replacement for efficient vision transformer, 2024.

Bin Lin, Yunyang Ge, Xinhua Cheng, Zongjian Li, Bin Zhu, Shaodong Wang, Xianyi He, Yang Ye, Shenghai Yuan, Liuhan Chen, Tanghui Jia, Junwu Zhang, Zhenyu Tang, Yatian Pang, Bin She, Cen Yan, Zhiheng Hu, Xiaoyi Dong, Lin Chen, Zhang Pan, Xing Zhou, Shaoling Dong, Yonghong Tian, and Li Yuan. Open-sora plan: Open-source large video generation model, 2024.

Enzhe Lu, Zhejun Jiang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Weikang Meng, Yadan Luo, Xin Li, Dongmei Jiang, and Zheng Zhang. Polaformer: Polarity-aware linear attention for vision transformers. In *International Conference on Learning Representations (ICLR)*, 2025.

Tobias Christian Nauen, Sebastian Palacio, Federico Raue, and Andreas Dengel. Taylorshift: Shifting the complexity of self-attention from squared to linear (and back) using taylor-softmax. *arXiv preprint arXiv:2403.02920*, 2024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations (ICLR)*, 2021.

Xiangyu Peng, Zangwei Zheng, Chenhui Shen, Tom Young, Xinying Guo, Binluo Wang, Hang Xu, Hongxin Liu, Mingyan Jiang, Wenjun Li, Yuhui Wang, Anbang Ye, Gang Ren, Qianran Ma, Wanying Liang, Xiang Lian, Xiwen Wu, Yuting Zhong, Zhuangyan Li, Chaoyu Gong, Guojun Lei, Leijun Cheng, Limin Zhang, Minghao Li, Ruijie Zhang, Silan Hu, Shijie Huang, Xiaokang Wang, Yuanheng Zhao, Yuqi Wang, Ziang Wei, and Yang You. Open-sora 2.0: Training a commercial-level video generation model in $200k, 2025.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosFormer: Rethinking softmax in attention. In *International Conference on Learning Representations (ICLR)*, 2022.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Winter Conference on Applications of Computer Vision (WACV)*, 2024.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. DriveLM: Driving with graph visual question answering. In *European Conference on Computer Vision (ECCV)*, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong

Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3d vision made easy, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yushu Wu, Zhixing Zhang, Yanyu Li, Yanwu Xu, Anil Kag, Yang Sui, Huseyin Coskun, Ke Ma, Aleksei Lebedev, Ju Hu, Dimitris Metaxas, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. SnapGen-V: Generating a five-second video within five seconds on a mobile device, 2025.

Ziyang Wu, Tianjiao Ding, Yifu Lu, Druv Pai, Jingyuan Zhang, Weida Wang, Yaodong Yu, Yi Ma, and Benjamin D. Haeffele. Token statistics transformer: Linear-time attention via variational rate reduction, 2024.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers, 2024.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025a.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025b.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer, 2025.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Jintao Zhang, Haoxu Wang, Kai Jiang, Shuo Yang, Kaiwen Zheng, Haocheng Xi, Ziteng Wang, Hongzhou Zhu, Min Zhao, Ion Stoica, Joseph E. Gonzalez, Jun Zhu, and Jianfei Chen. SLA: Beyond sparsity in diffusion transformers via fine-tunable sparse-linear attention, 2025.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*, 2024a.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *International Conference on Learning Representations (ICLR)*, 2024b. arXiv:2402.04347.

Lianghui Zhu, Zilong Huang, Bencheng Liao, Jun Hao Liew, Hanshu Yan, Jiashi Feng, and Xinggang Wang. Dig: Scalable and efficient diffusion models with gated linear attention, 2024.

Table 3: Composition of the synthetic training data at each stage.

Stage	Total	MQAR retrieval	Generic copy	Prose continuation
8k	**60,000**	22,500	7,500	30,000
16k	**30,000**	11,250	3,750	15,000
32k	**18,000**	6,750	2,250	9,000

A Datasets, Tasks, and Training Objectives

Here, we describe the data and objectives used in the final long-context curriculum.

A.1 BookSum

We derive long-context summarization examples from BookSum, a collection of literary texts paired with human-written summaries. Preprocessing first filters and deduplicates the source records and creates fixed train, development, and test splits. The second stage groups consecutive chapters from the same book into non-overlapping, length-budgeted windows. We use a curriculum with nominal context budgets of 8,192, 16,384, and 32,768 tokens. We arrive at 69960 training samples of length 8k, 28150 samples of length 16k and 10475 samples of length 32k.

For a tokenized dialogue $x_{1:T}$, the causal language-model loss is

$$\mathcal{L}(x) = -\frac{1}{|S|}\sum_{t\in S}\log p_\theta(x_t \mid x_{<t}),$$

where S contains only assistant-response positions. User tokens—including the instruction and source chapters—are visible as context but masked from the loss. For BookSum records, S is therefore the human summary. This prevents the much longer source text from dominating the objective. Summary cross-entropy should nevertheless be interpreted cautiously: a reference is one valid paraphrase among many, so its loss includes irreducible lexical and stylistic uncertainty. We retain it as the nominal task objective and a consistency metric, but use controlled retrieval evaluations for architectural claims.

A.2 Open-Vocabulary Associative Recall And Pose Continuation

We augment every curriculum stage with synthetic instruction-response records. Here, synthetic refers to the task construction and arbitrary payloads, not to the background language: the generator uses random contiguous token spans from the stream instruction and chapter text and and trims or concatenates it to the requested context budget. Training records use prose from the BookSum training split, whereas validation records use BookSum validation prose. It creates three record types: open-vocabulary multi-query associative recall (MQAR) records with several inserted key-value statements and a terminal query (see below); generic-copy records with one inserted arbitrary span to reproduce; and prose-continuation records with no inserted fact, for which the model predicts the remainder of the original passage. Each training set (8k, 16k or 32k) reserves 50% of its records for ordinary prose continuation. The remaining 50% is retrieval data, of which 25% is a generic copy subtask and 75% is MQAR. The resulting counts are listed in table 3.

Combining the data from the summarization and the MQAR task, we arrive at 129960, 58150, and 28475 samples per curriculum epoch at 8k, 16k, and 32k context length.

A.3 MQAR And Copy Tasks

For creating our MQAR data, we modify modify multi query associative recall (MQAR) (Arora et al., 2024).

We insert randomly generated key-value associations into the BookSum chapter text used as haystack. Each association is rendered as a short carrier sentence and inserted at a sampled depth among the original sentences of the long passage. A query is appended after the completed haystack, asking the model to reproduce the value associated with its key. The final recipe uses at most eight planted pairs, one query, and no additional distractors; the other planted pairs already serve as

Table 4: Examples of embedded carrier phrases and corresponding queries.

Embedded carrier phrase	Query
Remember: the RAMPART code is bI8_talen-49755.	What is the value of the RAMPART code?
Remember: the LKKGF code is wN5_brix-881.	What is the value of the LKKGF code?
Important: LANTERN maps to TIMBER-645.	Which code value belongs to LANTERN?
Important: DXRCBDJKV maps to YONTHOBREL-55082.	Which code value belongs to DXRCBDJKV?
For reference, the HARBOR code equals DEFENESTRATE-681.	State the value of the HARBOR code.
For reference, the JZWRKSXLB code equals SWFQD-5803.	State the value of the JZWRKSXLB code.

competing associations. The assistant answer is short (approximately 12–15 tokens), unique, and fully supervised, while the long prompt is masked. A low loss therefore requires locating the queried association over a long distance rather than merely modeling local prose.

A.4 Examples And Scale Of The Open-Vocabulary Codes

The value space is deliberately open and extremely large. Earlier data used a small closed label inventory and later used disjoint character inventories for training and evaluation; both admitted shortcuts. The final generator instead draws train and test values from the same character and subword inventory but from a combinatorial space large enough that accidental repetition is negligible. Consequently, token identity does not reveal the split, and memorizing a small answer vocabulary cannot solve the task. The generic copy subtask plants a single arbitrary value and requests it directly. It removes key-binding difficulty while preserving exact long-range copying, helping to distinguish failures of span copying from failures of associative binding.

The payload generator mixes several key generators rather than drawing answers from a small, fixed set. The measured training-side prefix space contains 24 color strings, 26 ordinary-word strings, 20 rare-word strings, 168,400 invented-word strings, approximately 5.6×10^9 random-string strings, and 162,240 mixed-case words. A numeric suffix is then added, producing keys such as:

- bI8_talen-49755
- sG8_brix-873
- TIMBER-645
- cresp_Je1-99651
- INGLENOOK-55
- THOMOLDUNSAR-78334
- PNNBMKH-847
- DEFENESTRATE-681

Here, a *value* is the complete answer code, consisting of a generated prefix plus a numeric suffix—for example, the value ‘TIMBER-645‘ has prefix ‘TIMBER‘ and suffix ‘645‘. Multiple values can therefore share a prefix while remaining different because their numeric suffixes differ (for example, ‘TIMBER-645‘ and ‘TIMBER-2719‘). In a sample of 20000 complete values, 19867 were unique (99.33%). Removing the numeric suffixes left 10852 distinct prefixes; 93.1% of those distinct prefixes appeared in only one of the 20000 draws, while the remainder were reused with different suffixes or, rarely, formed duplicate complete values. Likewise, drawing 100 keys produced 100 distinct keys. The measured overlap between complete train and test values was zero. These statistics make memorization of a compact answer inventory ineffective and force the model to copy the particular complete value embedded in the current context.

These are example of carrier phrases with embedded key-value pairs and possible query phrases:

A.5 Raw-Prose Next-Token Prediction

Purely synthetic retrieval training can preserve the puzzle-solving behavior while degrading general language modeling. We therefore devote half of each synthetic file to prose examples. A passage from the corresponding BookSum split (training prose for training files and development prose for validation files) is divided approximately 1:7: the first portion appears in the user prompt (*"Continue the following passage"*), and the remaining prose is the assistant target. Unlike summarization, the continuation has a unique observed target and places loss on thousands of natural-language tokens. It acts as an anti-forgetting regularizer and a diagnostic: rising held-out loss on prose continuation indicates degradation of ordinary language modeling even if retrieval loss improves. The generator draws filler from a large pool (3,000 source records in the final recipe) to avoid the prose reuse.

All records use the same token-level cross-entropy implementation, but per-example losses are assigned task-dependent multipliers. In the active configuration, the coarse BookSum summary group has weight 0.25, MQAR/copy has weight 0.75, and prose continuation has a weight of 1.5.

B Video Generation Results

B.1 Model Variants

We computed maximum attention scores for each self-attention block of the teacher model averaged over several batches. For **HLA-F2-21-480p**, we selected the 21 blocks (1,2,3,4,5,6,7,8, 11,12,13,14,15,16,17,18, 22,23,24,25,26) with the smallest scores. For **HLA-F3-15-240p**, we applied HLA to blocks 1,2,3,4,5,6,7,8, 11,12,13,14,15,16,17. For **HLA-F3-10-480p**, we only use HLA in the odd layers (1,3,5,7, 11,13,15,17, 23,25). We apply two small MLPs (see F) to the queries and attention outputs before element-wise multiplication.

B.1.1 Evaluation Protocol

We evaluate video-generation quality using VBench Huang et al. (2024). We report the aggregate VBench score and its component metrics to distinguish changes in perceptual quality, temporal consistency, motion quality, and text-video alignment.

B.1.2 Main Results

Table 5 reports video quality for the original Wan2.1-1.3B model, several prior works of comparable parameter count, a variant that replaced softmax attention by linear attention at 15 of its layers Ghafoorian et al. (2025), and the HLA-F3-10-480p variant. HLA-F2 achieves a VBench score of 81.4 compared with 78.9 for linear attention and 83.3 for the Wan baseline. At the same time, HLA-F3-10-480p requires 218 TFLOPs, corresponding to a reduction of almost 23% relative to the original model that requires 283 TFLOPs for a single forward pass through the model. Model HLA-F3-21-480p that replaces softmax attention in 21 out of 30 layers still achieves a VBench score almost 2 points higher than the linear attention variant that replaces softmax-attention in only 15 of its layers, see table 6.

The component-wise VBench results indicate that the proposed hybrid models reduce the necessary compute under a small impact to the quality of the generated videos. These findings show that higher-order attention is not restricted to autoregressive language modeling. Its successful integration into a video diffusion transformer indicates that the mechanism can operate under substantially different input structures, objectives, and generation procedures.

C Pseudocode

Algorithm 1 provides pseudocode of the entire attention procedure. PyTorch-like code for the implementation of the subroutines to compute two or three factor HLA can be found in figures 11 and 12.

	Total↑	Quality↑	Semantic↑
Models with at most 2B parameters			
Peng et al. (2025)	79.76	81.35	73.39
Wu et al. (2025)	81.14	83.47	71.84
HaCohen et al. (2024)	80.00	82.30	70.79
Yang et al. (2025b)	81.55	82.48	77.81
Isobe et al. (2025)	81.35	83.73	71.84
Huang et al. (2025)	81.91	83.36	76.10
Jin et al. (2025)	81.72	84.74	69.62
Wan et al. (2025)	83.31	85.23	75.65
Wan2.1-1.3B + linear attention, Ghafoorian et al. (2025)	78.9	82.2	65.9
Wan2.1-1.3B-HLA $\sim 23\%$ less total compute than Wan2.1-1.3B			
HLA-F3-10-480p	81.42	83.22	74.20

Table 5: Comparison on VBench with other models for video diffusion. We report the numbers published in the respective papers. While the proposed method has slightly lower scores, it about 23% less TFLOPs than Wan2.1-1.3B. The results for linear attention stem from Ghafoorian et al. (2025) who replaced self-attention in 15 transformer blocks by standard linear attention in Wan2.1-1.3B.

Method	Tokens	Score	TFLOPs	Time
Wan2.1 (quad.)	32760	83.3	283.03	1:36
Wan2.1 (quad.)	12600	81.0	62.13	0:45
HLA-F3-10-480p	32760	81.42	218.59	1:26
HLA-F3-15-240p	12600	79.78	48.37	0:26
HLA-F3-21-480p	32760	80.54	147.71	1:16

Table 6: VBench scores and times to generate a single video of several model variants. TFLOPs are for a single forward pass through the model. All measurement were done on an H100 GPU.

D AI Policy

E Ethics Statement

This work investigates architectural and training mechanisms for improving sequence modeling and associative recall. Our experiments primarily rely on established public data. For multi-query associate recall, we use synthetic data as established by prior works. Our experiments do not involve human subjects or personally identifiable information. The proposed methods are intended to advance the understanding of sequence models rather than to target a particular downstream application. As with general-purpose improvements to machine learning models, the techniques introduced here could potentially be incorporated into systems deployed in applications with societal impact. Such deployment may inherit risks associated with the underlying models and training data, including biases, unreliable predictions, and inappropriate use. These risks are not specific to the methods proposed in this work. We encourage appropriate evaluation and safeguards when applying the methods in high-impact settings.

F Model Cards

All models are derived from `Qwen2.5-1.5B-Instruct`. Unless stated otherwise, we preserve the pretrained query, key, value, and output projections, the rotary position embeddings, and the remaining transformer components. We replace only the self-attention module in every transformer layer. Newly introduced feature maps and gating projections are randomly initialized and trained during adaptation.

MLP Feature Map Several models use a learned per-head feature map $\phi : \mathbb{R}^{d_h} \to \mathbb{R}^{d_\phi}$ of the form

$$\phi(\mathbf{x}) = \mathrm{softplus}(\mathbf{W}_2\,\mathrm{GELU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)\,, \tag{11}$$

Algorithm 1 Hadamard Linear Attention

Input: Input tokens X
Output: Attention output

Step 1: Generate queries, keys, values
$q, k, v \leftarrow \texttt{to_q}(X), \texttt{to_k}(X), \texttt{to_v}(X)$
reshape tensors to allow for multi-head attention

Step 2: Apply rotational embedding
$q, k \leftarrow \texttt{apply_rope}(q), \texttt{apply_rope}(k)$

Step 3: Apply scale factor
$q \leftarrow q \cdot \texttt{scale_factor}$

Step 4: HLA-specific transformations
$q \leftarrow \texttt{hla_to_q}(q)$
$k_1, k_2 \leftarrow \texttt{hla_to_k1}(k), \texttt{hla_to_k2}(k)$
if `use_hla_3_factors` **then**
  $k_3 \leftarrow \texttt{hla_to_k3}(k)$
end if

Step 5: Compute HLA
if `use_hla_2_factors` **then**
  # by algorithm in Fig. 4 (supplementary)
  $\texttt{attn} \leftarrow \texttt{2_factor_hla}(q, k_1, k_2, v)$
end if
if `use_hla_3_factors` **then**
  # by algorithm in Fig. 5 (supplementary)
  $\texttt{attn} \leftarrow \texttt{3_factor_hla}(q, k_1, k_2, k_3, v)$
end if

Return `to_out(attn)`

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{lat}} \times d_h}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_\phi \times d_{\text{lat}}}$. The softplus output activation ensures that the resulting linear-attention features are strictly positive. Both weight matrices are initialized uniformly from $\mathcal{U}(-0.1, 0.1)$, and both biases are initialized to zero. No input layer normalization or intermediate output ReLU is used in the configurations below.

Softmax Attention Baseline The baseline is the unmodified self-attention mechanism of `Qwen2.5-1.5B-Instruct`. It uses the pretrained query, key, value, and output projections together with rotary position embeddings and grouped-query attention. This model serves as the dense quadratic-attention reference.

Symmetric HLA-F2 Hadamard Linear Attention with two factors, denoted HLA-F2, replaces softmax attention with a causal factorized linear-attention mechanism. After the pretrained query and key projections and rotary position embeddings, we compute

$$\mathbf{Q} = \phi_q(\mathbf{q}), \qquad \mathbf{K}_1 = \phi_{k_1}(\mathbf{k}), \qquad \mathbf{K}_2 = \phi_{k_2}(\mathbf{k}), \tag{12}$$

using three independently parameterized MLP feature maps. The corresponding second-order query feature is symmetric, whereas the key feature is represented by the Hadamard-factorized pair $(\mathbf{K}_1, \mathbf{K}_2)$. The causal HLA-F2 operator is evaluated using a custom CUDA kernel. We use $d_{\text{lat}} = 16$ and $d_\phi = 16$. The difference to the operator proposed in Arora et al. (2024); Nauen et al. (2024) is that we use untied featuer maps for $\mathbf{K}_1$ and $\mathbf{K}_2$.

Asymmetric HLA-F2 The asymmetric HLA-F2 variant learns two independent query factors and two independent key factors:

$$\mathbf{Q}_1 = \phi_{q_1}(\mathbf{q}), \qquad \mathbf{Q}_2 = \phi_{q_2}(\mathbf{q}), \qquad \mathbf{K}_1 = \phi_{k_1}(\mathbf{k}), \qquad \mathbf{K}_2 = \phi_{k_2}(\mathbf{k}). \tag{13}$$

Thus, both the query and key addressing features are represented by factorized second-order feature maps. Unlike symmetric HLA-F2, the two query factors are parameterized independently. We use $d_{\text{lat}} = 128$ and $d_\phi = 64$ for each factor.

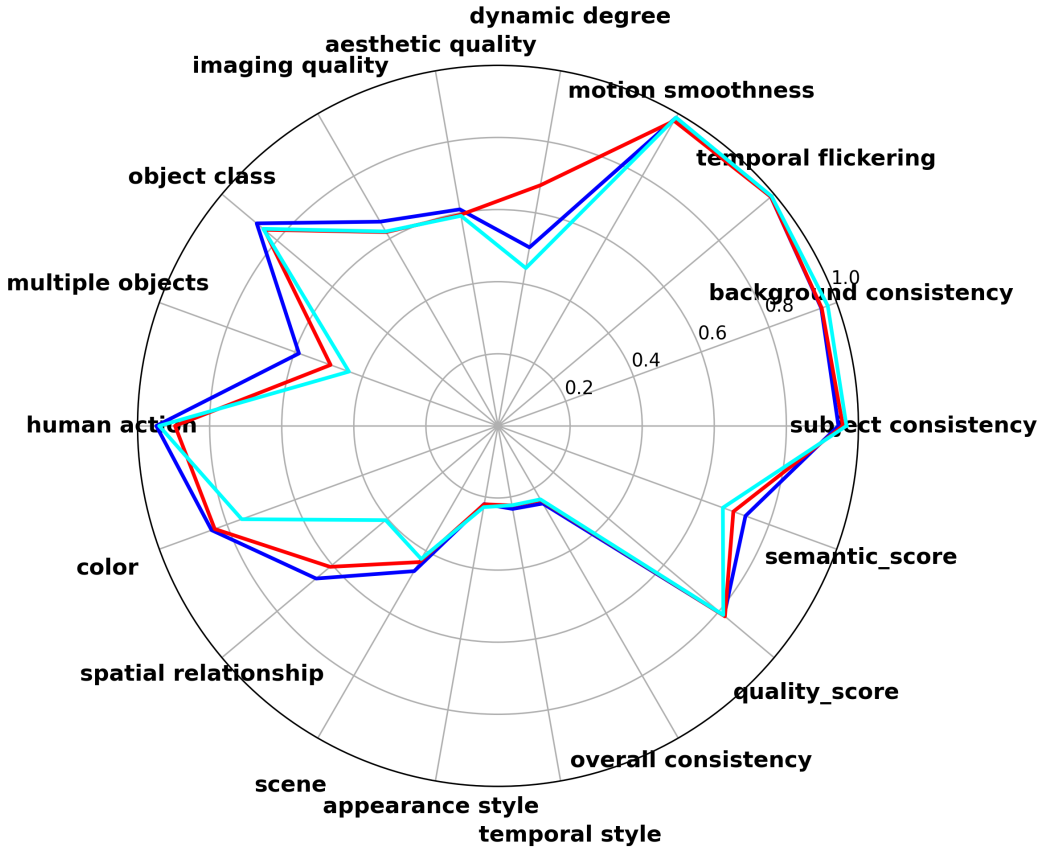

Figure 3: VBench scores by categories. Dark blue line corresponds to **HLA-3F-R1-10**, red to **HLA-3F-R1-21** and light blue to **HLA-3F-R2-15**.

HOLA-F3 This model combines first-, second-, and third-order causal linear-attention operators. The first-order query and key features are obtained using full-width projections over the complete hidden representation:

$$\mathbf{Q}^{(1)} = \mathrm{ReLU}\left(\mathbf{W}_q^{(1)}\mathbf{q}\right), \qquad \mathbf{K}^{(1)} = \mathrm{ReLU}\left(\mathbf{W}_k^{(1)}\mathbf{k}\right). \tag{14}$$

The first-order components are subtracted from the per-head query and key representations. Independent MLP feature maps are then applied to the resulting residuals to construct symmetric second- and third-order features. The second-order feature maps have width d_ϕ, while the third-order feature maps have width $d_\phi/2$. The outputs $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$ of the three causal attention operators are combined using input-dependent mixture weights:

$$\boldsymbol{\gamma} = \mathrm{softmax}\left(g\left(\frac{1}{N}\sum_{t=1}^{N}\mathbf{q}_t\right)\right), \qquad \mathbf{U} = \sum_{r=1}^{3}\gamma_r\mathbf{U}^{(r)}, \tag{15}$$

where g is a per-head two-layer MLP producing three gate logits. We use $d_{\mathrm{lat}} = 128$ and $d_\phi = 16$, giving a third-order factor width of 8.

Sink Sliding-Window Attention Sink sliding-window attention Jolicoeur-Martineau et al. (2026) retains exact softmax attention but restricts the causal attention pattern. Each query attends to the first N_{sink} tokens and to the most recent W tokens:

$$\mathcal{A}(t) = \{0, \ldots, N_{\mathrm{sink}} - 1\} \cup \{\max(0, t - W + 1), \ldots, t\}. \tag{16}$$

We use a sliding-window size of $W = 512$ and $N_{\mathrm{sink}} = 4$ sink tokens. All pretrained attention projections are preserved. For this model, we apply LoRA with rank 16, scaling parameter 32,

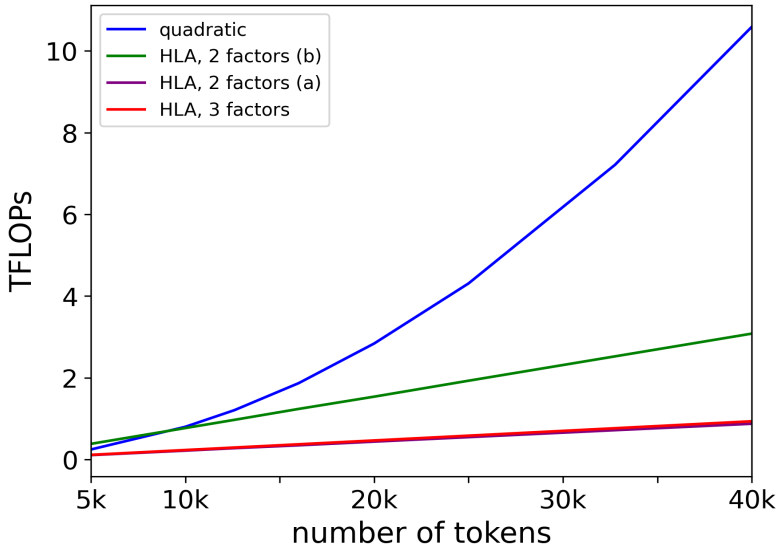

Figure 4: Comparison of computational complexities for different sequence lengths between standard quadratic attention and 2 or 3-factor HLA. HLA 2 factors (a) uses small MLPs with identical parameters as HLA 3 factors whereas HLA 2 factors (b) uses larger MLPs. As can be seen, the curves for HLA 3 factors and HLA 2 factors (a) almost coincide. This implies that in the regime of large number of tokens and between 2 and 3 Hadamard factors, the size of the MLPs is decisive.

and dropout 0.05 to the attention projections (`q_proj`, `k_proj`, `v_proj`, and `o_proj`) and the feed-forward projections (`gate_proj`, `up_proj`, and `down_proj`). We train LoRA biases only and do not use rank-stabilized LoRA or DoRA.

Chunked Gated Delta Net Chunked Gated DeltaNet replaces softmax attention with a linear-recurrent delta-rule update. It preserves the pretrained query, key, value, and output projections and introduces two projections that produce one decay gate and one write-strength gate per attention head and token:

$$\alpha_t = \sigma(\mathbf{W}_\alpha \mathbf{h}_t + \mathbf{b}_\alpha)\,, \qquad \beta_t = \sigma(\mathbf{W}_\beta \mathbf{h}_t + \mathbf{b}_\beta)\,. \tag{17}$$

Here, α_t controls retention of the recurrent state, while β_t controls the strength of the delta-rule update. Computation is organized into chunks of 512 tokens. The gate biases are initialized to $b_\alpha = 4.0$ and $b_\beta = 0.0$, corresponding to initial gate values $\sigma(4.0) \approx 0.982$ and $\sigma(0) = 0.5$, respectively.

Asymmetric HLA-F2 Gated Delta Net This model combines the chunked gated delta rule with the addressing space of asymmetric HLA-F2. Instead of using first-order query and key features, it constructs two independent positive factors for each:

$$\boldsymbol{\phi}_q(\mathbf{q}) = \phi_{q_1}(\mathbf{q}) \otimes \phi_{q_2}(\mathbf{q}), \qquad \boldsymbol{\phi}_k(\mathbf{k}) = \phi_{k_1}(\mathbf{k}) \otimes \phi_{k_2}(\mathbf{k}), \tag{18}$$

where $\otimes$ denotes the factorized outer-product feature representation. These features are used for addressing within the gated delta-rule recurrence. The model otherwise uses the same per-token decay and write-strength gates as Chunked Gated Delta Net. We use a chunk size of 512, $b_\alpha = 4.0$, $b_\beta = 0.0$, $d_{\text{lat}} = 128$, and $d_\phi = 64$ for each Hadamard factor.

G Use of LLMs

LLM tools were used for software-development support, including the creation and optimization of CUDA kernels, the development of theoretical and conceptual frameworks, the formulation of mathematical claims, hypothesis generation and refinement, feedback on research methodology and experimental design and drafting, revising, and improving the readability of the manuscript.

All AI-generated outputs were reviewed and, where necessary, modified by the authors. Generated code was evaluated using reference implementations and numerical or experimental tests. Mathematical claims, theoretical suggestions, hypotheses, and methodological recommendations were independently assessed by the authors. AI-assisted text, factual claims, and references were reviewed for accuracy, originality, and appropriate attribution. The authors made all final research decisions and take full responsibility for the content of the paper.

H QUALITATIVE EXAMPLES

We present further qualitative examples of all three variants in Figs. 8, 9 and 10. As can be seen, all three methods achieve to generate high-quality videos. Please also see the original mp4 files in the supplementary material.



Figure 5: Examples of generated videos by **HLA-3F-R1-10**.

I CODE LISTINGS

We provide PyTorch-like code in listings 11 and 12. These algorithms do not require time-consuming tensor reshaping. They only require elementary computational operations.

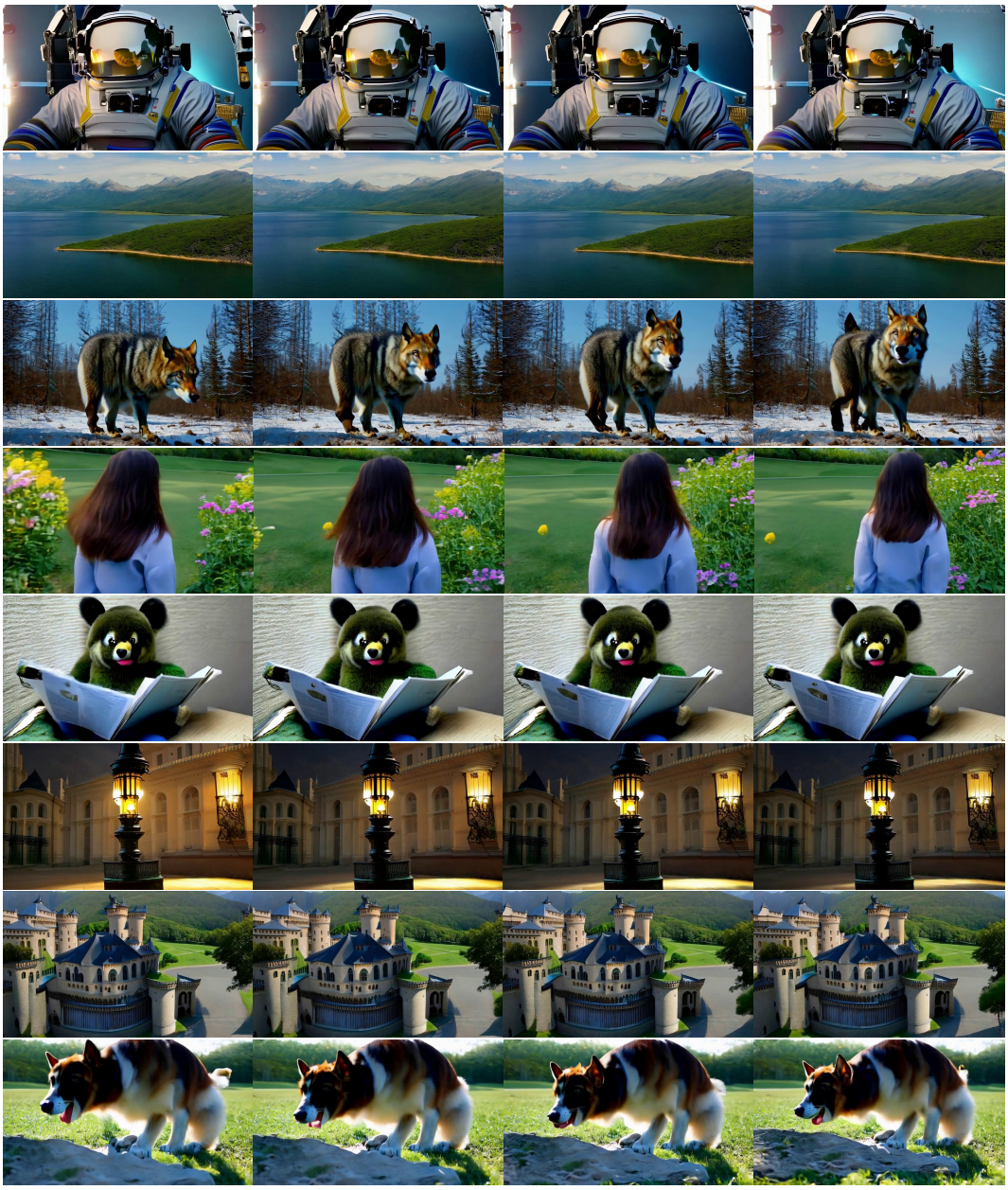

Figure 6: Examples of generated videos by **HLA-3F-R1-21**.

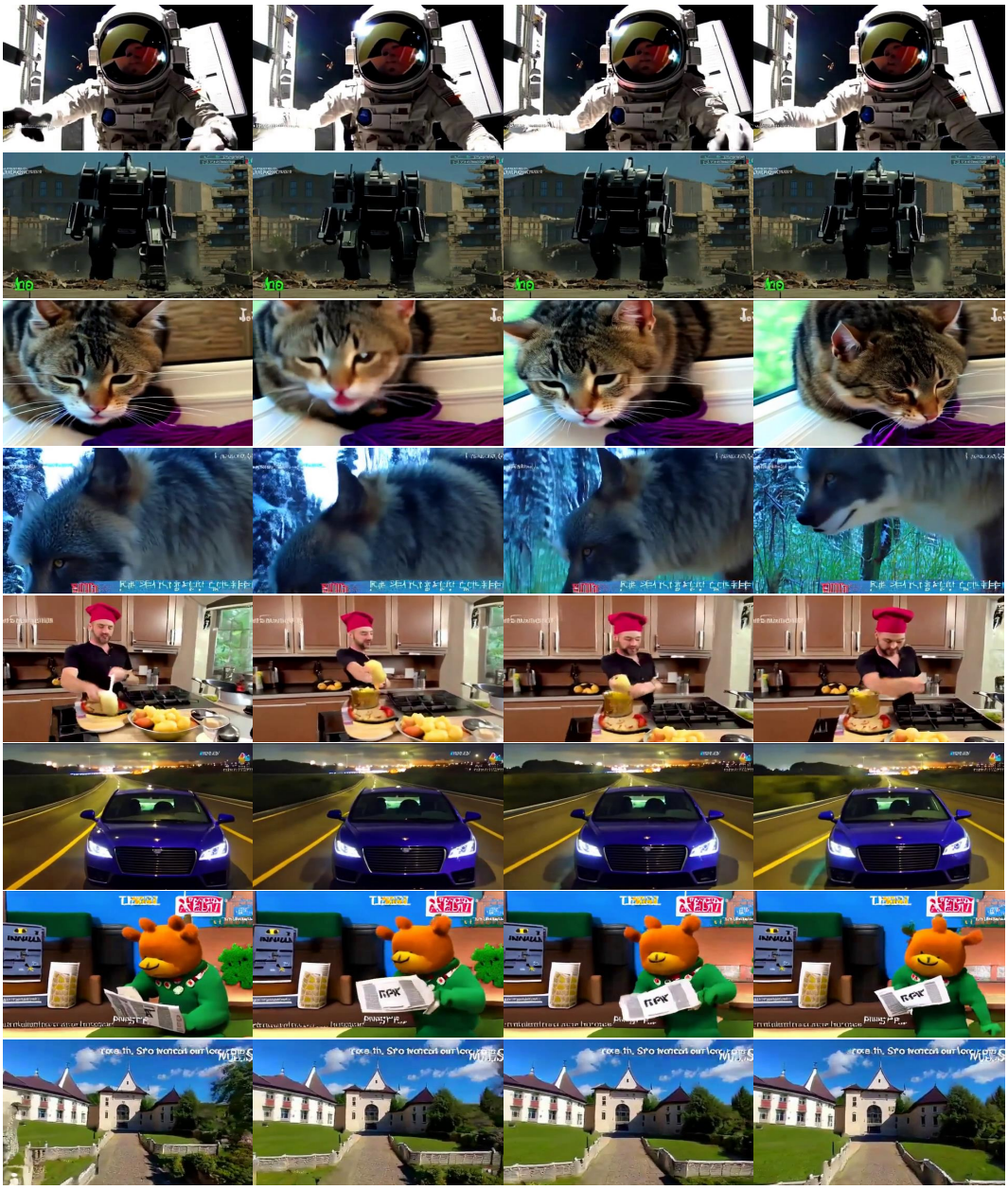

Figure 7: Examples of generated videos by **HLA-3F-R2-15**.



Figure 8: Qualitative Examples **HLA-3F-R1-10**



Figure 9: Qualitative Examples **HLA-3F-R1-21**



Figure 10: Qualitative Examples **HLA-3F-R2-15**

```python
def HLA_2_factors(query, key1, key2, value):
    # equivalent to ((Q@K1.t()) * (Q@K2.t())) @ V
    # (omitting normalization for simplicity)

    # batch size, number of heads, sequence length, d_phi
    bs, nh, sl, dp = query.shape
                 d = value.shape[-1]

    # compute outer products between query and key1, key2 tokens
    k_outerprod_k = torch.einsum('bhsd,bhse->bhsde',key1, key2)
    context       = torch.einsum(
                        'bhmde,bhmg->bhdeg',
                        k_outerprod_k, value
                    )
    q_outerprod_q = torch.einsum('bhsd,bhse->bhsde',query, query
        )

    # normalization
    sum_over_k    = k_outerprod_k.sum(dim=2, keepdim=True)
    eta           = torch.sum(
                        q_outerprod_q.view(bs,nh,sl,-1)
                        *
                        sum_over_k.view(bs,nh,1,-1),
                        dim=-1, keepdim=True
                    )
    q_outerprod_q = q_outerprod_q / (eta.view(bs,nh,sl,1,1)+eps)

    # memory efficient computation
    q_outerprod_q = q_outerprod_q.view(bs, nh, sl, dp**2)
    context       = context.view(bs, nh, dp**2, d)

    # output of linear attention (before ouput projection)
    attention     = torch.einsum(
                        'bhsd,bhde->bhse',
                        q_outerprod_q, context
                    )

    return attention
```

Figure 11: PyTorch-like code of Hadamard Linear Attention with 2 factors.

```
def HLA_3_factors(query, key1, key2, key3, value):
    # equivalent to ((Q@K1.t()) * (Q@K2.t()) * (Q@K3.t())) @ V
    # (omitting normalization for simplicity)

    # batch size, number of heads, sequence length, d_phi
    bs, nh, sl, dp = query.shape
                 d = value.shape[-1]

    # compute outer products between query and key1, key2 tokens
    k_outerprods  = torch.einsum(
                        'bhsd,bhse,bhsf->bhsdef',
                        key1, key2, key3
                    )
    context       = torch.einsum(
                        'bhmdef,bhmg->bhdefg',
                        k_outerprods, value
                    )
    q_outerprods  = torch.einsum(
                        'bhsd,bhse,bhsf->bhsdef',
                        query, query, query
                    )

    # normalization
    sum_over_k    = k_outerprods.sum(dim=2, keepdim=True)
    eta           = torch.sum(
                        q_outerprods.view(bs,nh,sl,-1)
                        *
                        sum_over_k.view(bs,nh,1,-1),
                        dim=-1, keepdim=True
                    )
    q_outerprods  = q_outerprods / (eta.view(bs,nh,sl,1,1,1)+eps)

    # memory efficient computation
    q_outerprods  = q_outerprods.view(bs, nh, sl, dp**3)
    context       = context.view(bs, nh, dp**3, d)

    # output of linear attention (before ouput projection)
    attention     = torch.einsum(
                        'bhsd,bhde->bhse',
                        q_outerprods, context
                    )

    return attention
```

Figure 12: PyTorch-like code of Hadamard Linear Attention with 3 factors.